Continuous Assessments

CA – 30%, Exam – 70%

Two lab tests (Prolog)

- Week 6 – Lab test 1 (15%)
- Week 11 – Lab test 2 (15%)

RULE-BASED EXPERT SYSTEMS

Negnevistsky, M. "Artificial Intelligence: A guide to Intelligent Systems" – chapter 2

Lecture overview

Types of knowledge

Rule-based expert systems

- Architecture
- Structure of a rule
- Inference - executing rules, conflict resolution, forward/backward chaining
- Advantages/disadvantages

Expert systems

What is intelligence?

Intelligence could mean the ability to

- Learn from new situations
- Apply solutions of old problems to new situations
- Communicate about a topic (even not familiar one)
- Apply abstract reasoning
- Understand how world works in general

General vs. Domain-specific knowledge

General purpose intelligence requires vast amount of knowledge about the world.

- *"Sorry but the bank was closed! Would you mind getting this round?"*

Very hard to collect, to represent and to use.

Domain-specific knowledge

Restricted to particular domain

- e.g. migraines, risk assessment, etc.

Knowledge is deep but not wide

- e.g. migraines but not other medical problems

Knowledge is tied directly to problem solving

- e.g. *diagnosis* of migraines, not their social repercussions

Much more feasible for implementation than the general knowledge

Procedural vs. prepositional

Most programming languages are **procedural**

- first do this, then do that…

Most human expert knowledge is **propositional**

- **If** [*you have a temperature*] **then** [*you have a cold*]…

Quantitative vs. qualitative

Quantitative representation

deals with exact values

Qualitative

is in general more subjective to interpretation

For example, representing height

- Qualitative approach

  John is tall, but Jim is taller.

- Quantitative approach

  John's height is 6'5'', and Jim's height is 6'8''.

Heuristic vs. rigid rules

Rigid rules

"If you have more than 10 coughs per hour then you have bronchitis."

Heuristics

"Frequent coughing means bronchitis."

What is an Expert System?

An expert system is a computer system that

- Performs functions similar to those normally performed by a human expert
- Uses a representation of human expertise in a specialist domain to perform functions similar to a human domain expert
- Operates by applying an inference mechanism to a body of specialist expertise represented in a knowledge base

Explanation

A human expert is usually able to explain their reasoning.

"Why do you think I have a migraine?"

"You have frequent, intense pain in the temple area, associated with nausea. Also you are not taking any medication that is likely to cause these side-effects. Therefore I am pretty sure your symptoms are a sign of a migraine."

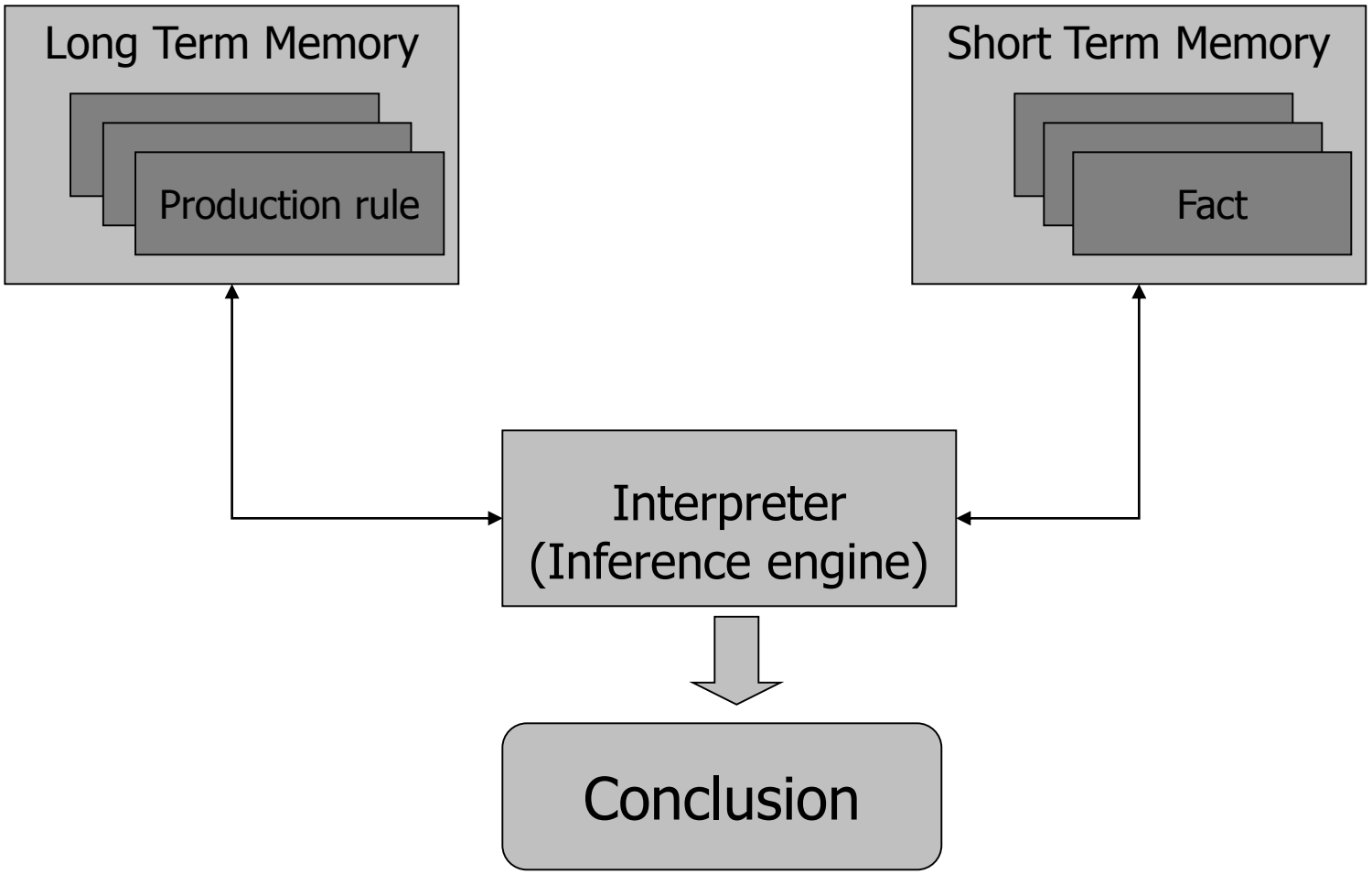
Rule-based systems

Rule-based systems (production systems) represent knowledge in terms of a set of rules that tell you what you should do or what you could conclude in different situations.

A rule-based system consists of

- a set of IF-THEN ***rules (production rules)***
- a set of ***facts***
- an ***interpreter*** controlling the application of the rules, given the facts.

Rule-based system model



Production rules

IF: the traffic light is green
THEN: the action is go

IF: the traffic light is red
THEN: the action is wait

Production rules

IF

<antecedent>

THEN

<consequent>

antecedent (premise or condition)

relates to what the system already knows.

consequent (conclusion or action)

relates to the conclusions the system may reach should the antecedent be satisfied.

A rule is **triggered** when its condition part is satisfied and this caused the action part to be executed (the rule **fires**). Several rules can be triggered, usually one is fired.

Production Rules (cont.)

Rule can have multiple antecedents

- Conjunction AND

IF <antecedent$_0$> AND <antecedent$_1$> … AND <antecedent$_n$> THEN <consequent>

- Disjunction OR

IF <antecedent$_0$> OR <antecedent$_1$> … OR <antecedent$_n$> THEN <consequent>

- Or a combination of both

IF <antecedent0> AND <antecedent1> … AND <antecedentn> OR <antecedent$_0$> OR <antecedent$_1$> … OR <antecedent$_n$> THEN <consequent>

Production rules - Example

IF

animal is horse shaped ***AND*** *animal has stripes*

THEN

animal is zebra

IF

animal is lion ***OR*** *animal is tiger*

THEN

animal is dangerous

Production Rules (cont…)

Consequents can also have multiple clauses

IF <antecedent> THEN <consequent$_0$>, <consequent$_1$>,
…<consequent$_{n-1}$>, <consequent$_n$>

More production rule examples:

IF	**the fuel tank is empty**
THEN	**the car is dead**

IF	**the season is autumn**
AND	**the sky is cloudy**
AND	**the forecast is drizzle**
THEN	**advice is take an umbrella**

IF	**patient has stomach pains**
AND	**(temperature > 98**
OR	**patient is nauseous)**
THEN	**diagnosis is appendicitis,**
	action is call the surgeon

Inference engine

The inference engine is responsible for the application of the rules to the facts.

- This process results in the inference of new facts about the given problem.

Inference engines work in two stages:

- They identify the set of rules that may be triggered at a specific time.
- A particular rule is selected from this set of rules and executed.
- The process terminates when there are no more rules that could be triggered, or when a goal is satisfied.

The identification of which rules have been triggered takes one of two forms:

- either all the rules are evaluated, or
- only the rules that have not been triggered previously are evaluated.

Interpretation of a rule

When a rule is triggered, the inference engine causes the knowledge-base to be updated in some way

The interpretation of a rule dictates the way in which the facts inferred using the rule are placed into the working memory.

There are basically two main interpretations in the area of production systems.

- Interpreting as ADD.
  - inferred facts are simply added to the working memory.
- Interpreting as UPDATE.
  - inferred facts replace those facts that caused the rule to be triggered

Example

```
IF      hot AND sunny
THEN    good_day
```

Two ways to interpret this rule:

```
IF      hot AND sunny
THEN
        ADD good_day to the KB
```

```
IF      hot AND sunny
THEN
        REPLACE hot AND sunny
        WITH good_day in the KB
```

Forward vs. backward chaining

In a **forward chaining** system (**data-driven** approach) we start with the initial facts, and keep using the rules to draw new conclusions (or take certain actions) given those facts.

In a **backward chaining** system (**goal-directed** approach) we start with some hypothesis (or goal) we are trying to prove, and keep looking for rules that would allow us to conclude that hypothesis, perhaps setting new sub-goals to prove as we go.

Forward chaining

Take the known facts (from the database) and try to match against the antecedents of rules from the rulebase.

- If a rule is fired, then the consequents are added to the database.
- This system deduces new facts based on existing facts and the knowledge of the problem domain.
- Inference engines that use the forward chaining strategy apply the strategy exhaustively, until no new deductions are made.
- It is also known as **data-driven inference**.

Backward chaining

Take the facts (goals) and try to match against the consequents of rules from the rulebase.

- If a rule is triggered (or fired), then the antecedents are added to the database, and the consequents are removed.
- Inference engines that use the backward chaining strategy apply the strategy exhaustively, until no more rules are triggered.
- This system deduces what must be true for some goal(s) to hold.
- It is also known as **goal-directed** inference.

Backward chaining

Backward chaining informs us of the correctness (truthfulness) of some hypothesis (goal).

Backward chaining systems include a **goal set**. These are the hypotheses we are trying to prove.

The purpose of backward chaining is to prove some goal (G).

- If G is in the initial facts (database) it is said to be proven.
- Otherwise find a rule that may be used to conclude G, and try to prove each of that rule's premises. G is proved true if all the premises are proved true.

Forward vs. backward chaining

Study how a domain expert solves a problem

- If expert needs to gather information then infers from it
  → Choose forward chaining

- If expert beings with hypothesis and attempts to find facts to prove it
  → Choose backward chaining

Forward chaining is a natural way to design analysis and interpretation systems

Backward chaining is a natural way to design diagnostic systems

Example

Rules:

R1: **IF** hot AND smoky **THEN** ADD fire
R2: **IF** alarm_beeps **THEN** ADD smoky
R3: **IF** fire **THEN** ADD switch_on_sprinklers

Facts:

F1: alarm_beeps
F2: hot

Inference engine – forward chaining

- Triggers R2, adds F3: smoky
- Triggers R1, adds F4: fire
- Triggers R3, adds F5: switch_on_sprinklers

Example (cont.)

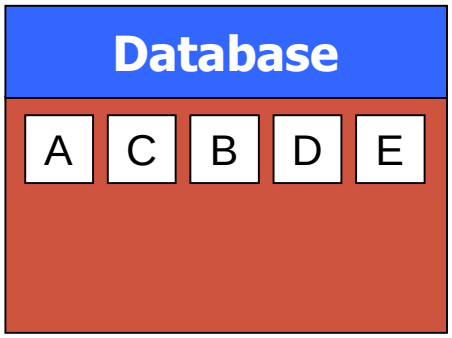
Inference engine – backward chaining

- G1: switch_on_sprinklers ?

- Triggers R3, adds G2: fire
- G2 → Triggers R1, adds G3: hot , G4: smoky
- G3 in initial facts
- G4 → Triggers R2, adds G5: alarm_beeps
- G5 in initial facts. **END**.

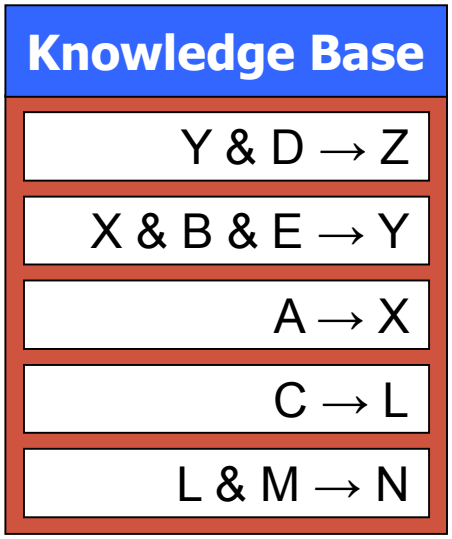
- Proved hypothesis **switch_on_sprinklers**.

Forward Chaining Example

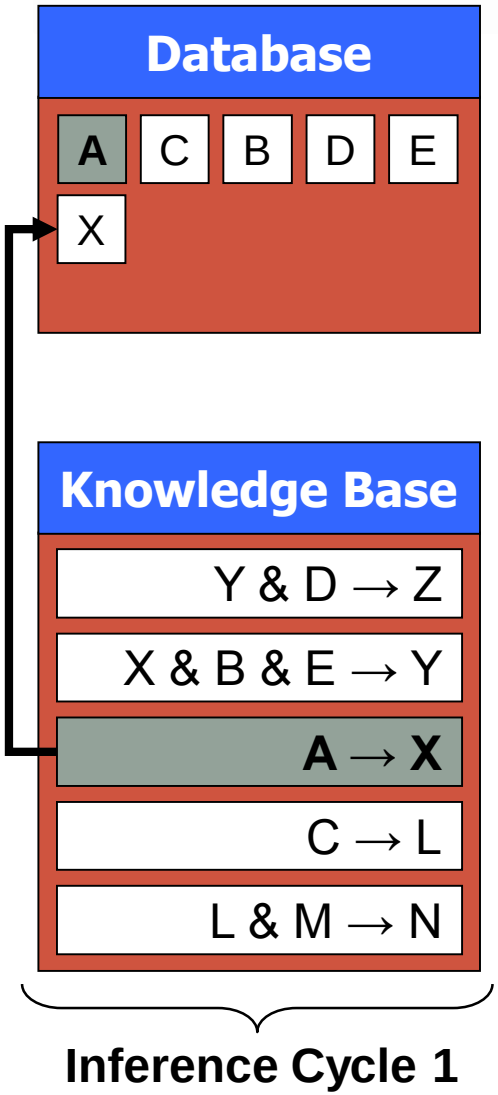


Suppose that we know the facts A, B, C, D, E and the rules shown in the knowledge base to the left

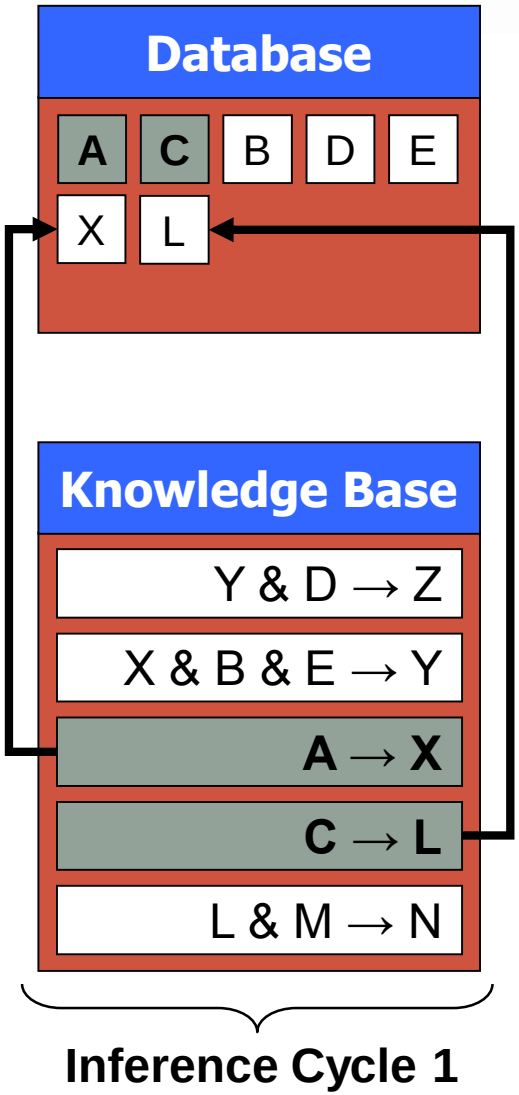
What facts can we infer from this?



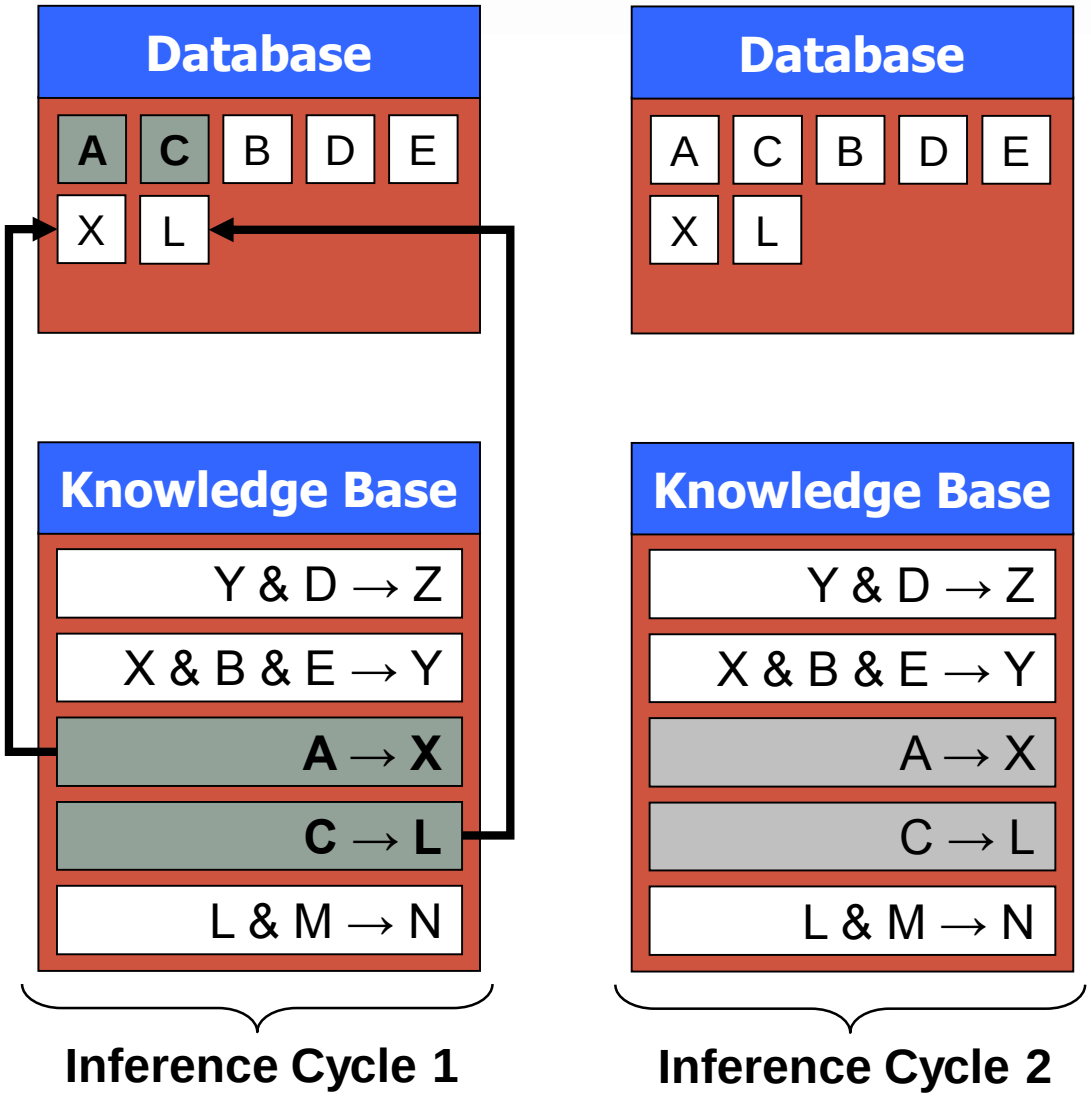
Forward Chaining Example



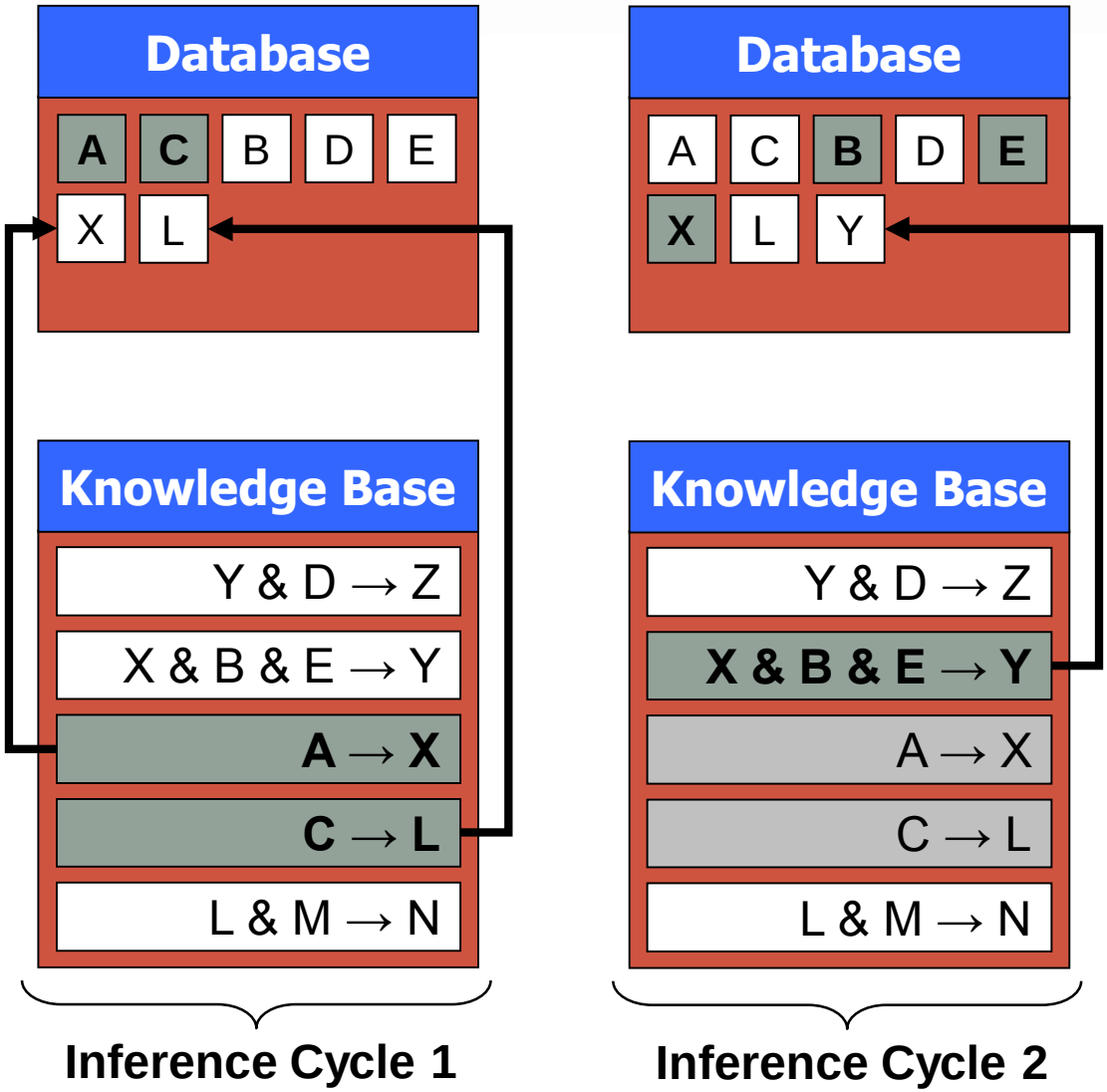
Forward Chaining Example



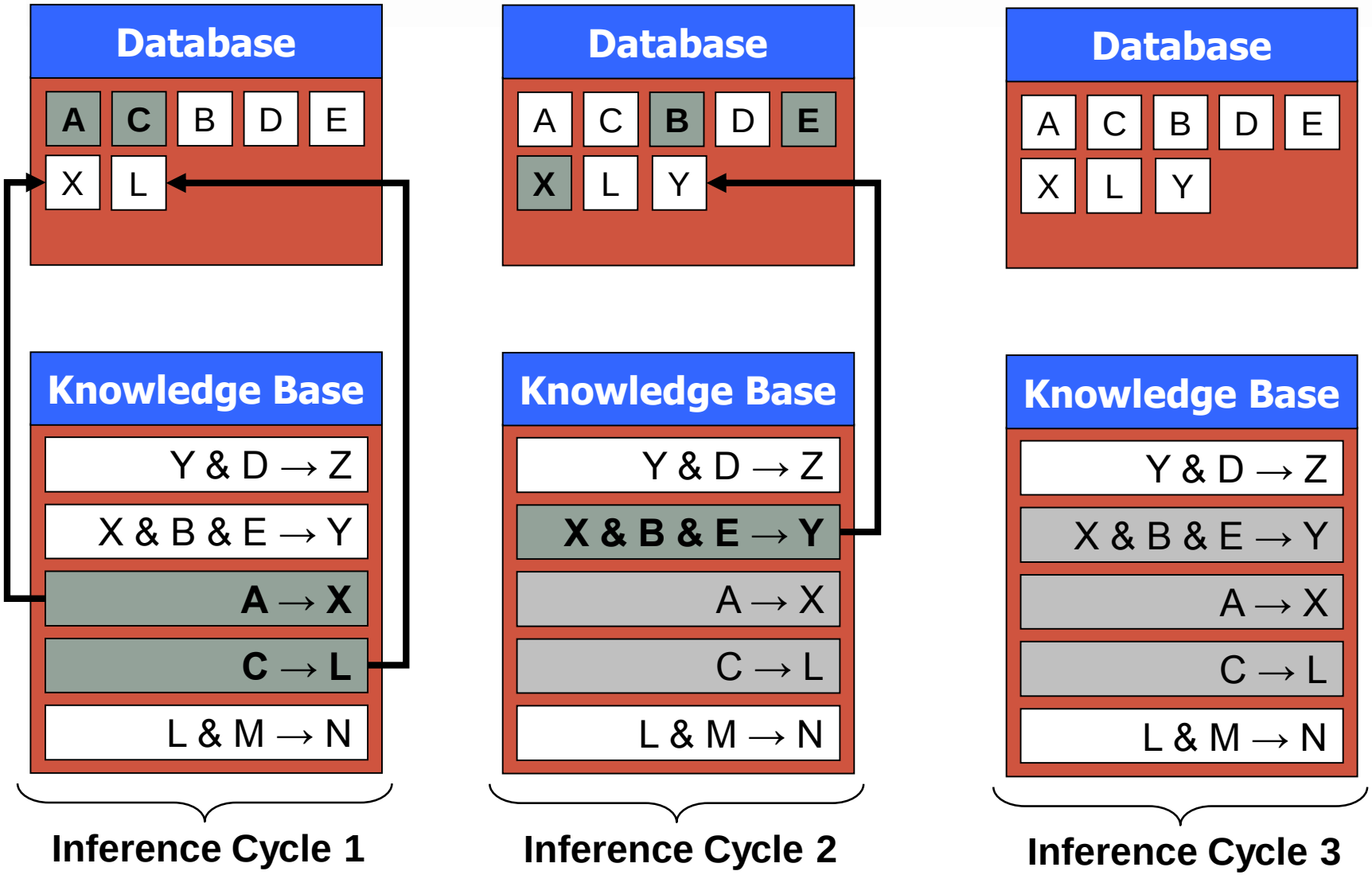
Forward Chaining Example



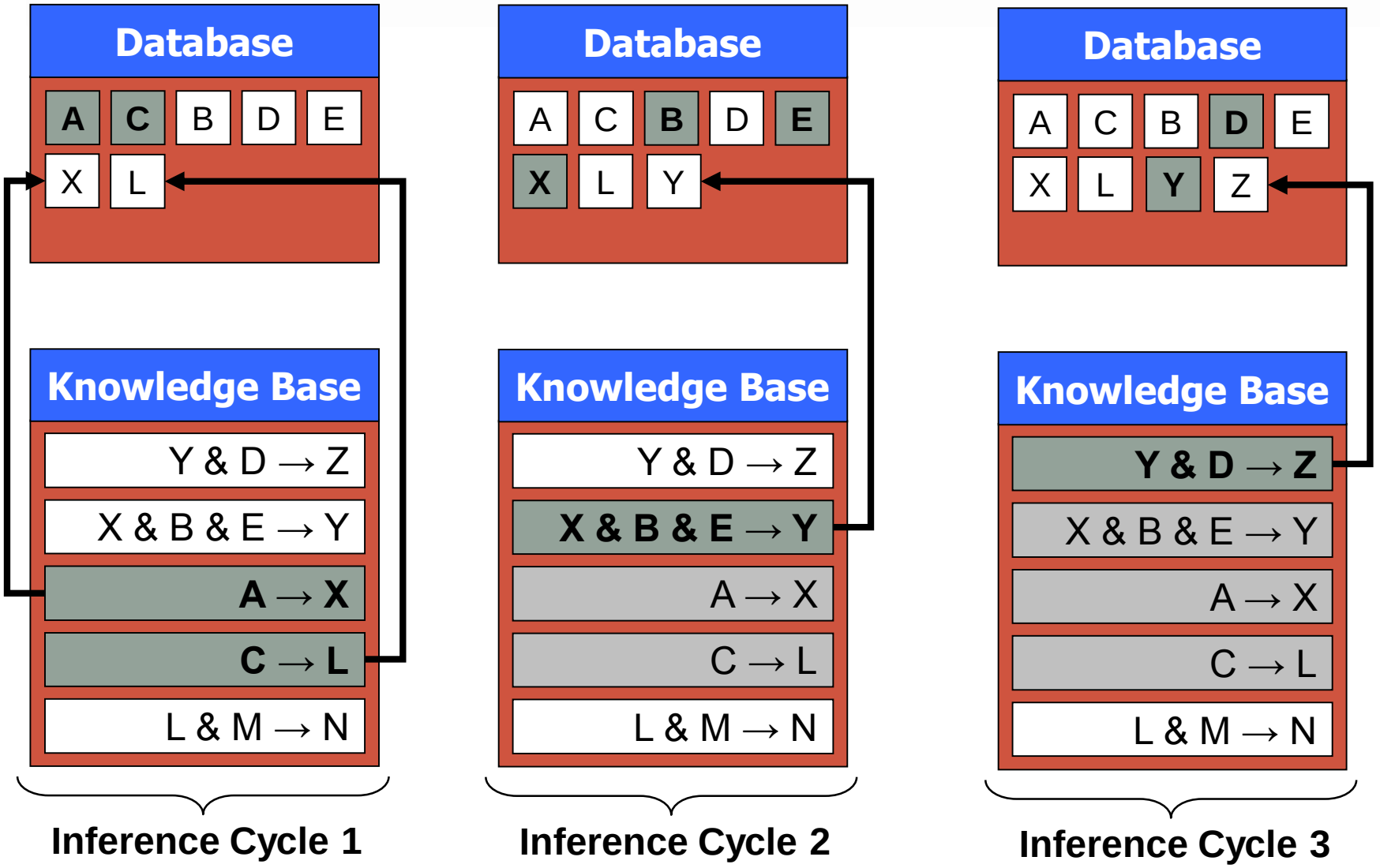
Forward Chaining Example



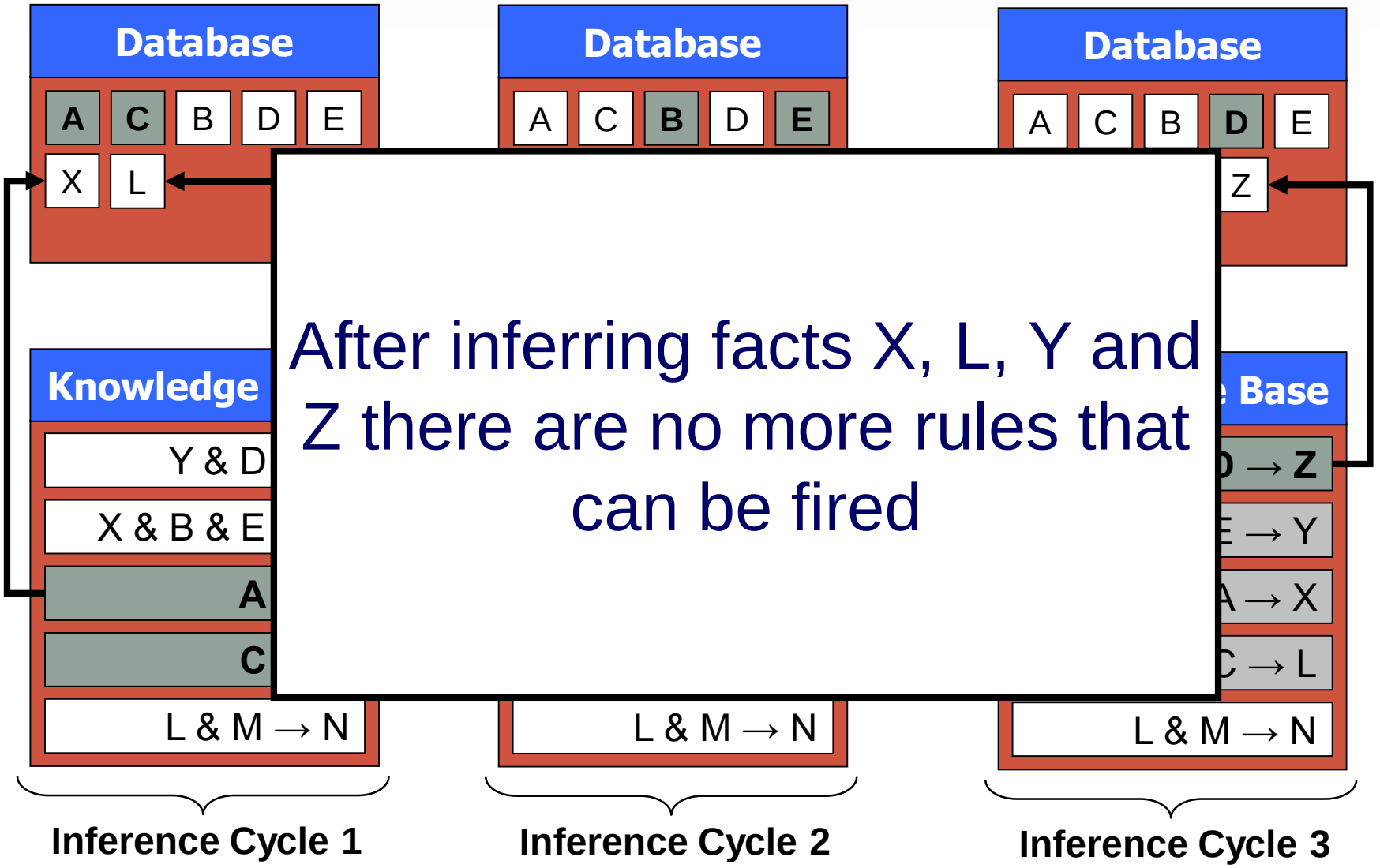
Forward Chaining Example



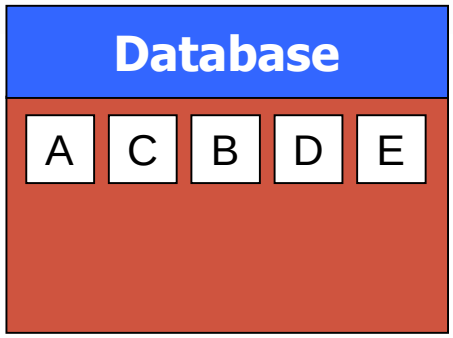
Forward Chaining Example



Forward Chaining Example

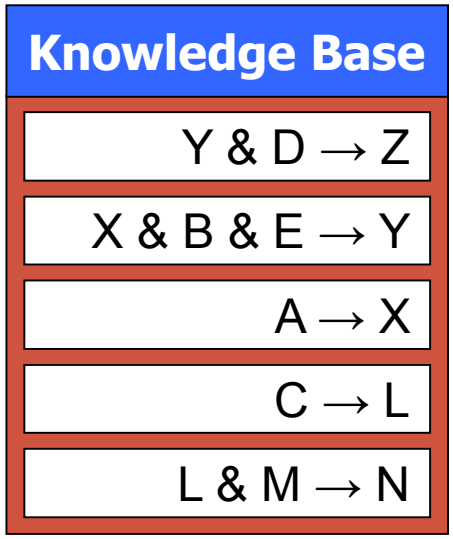


Backward Chaining Example

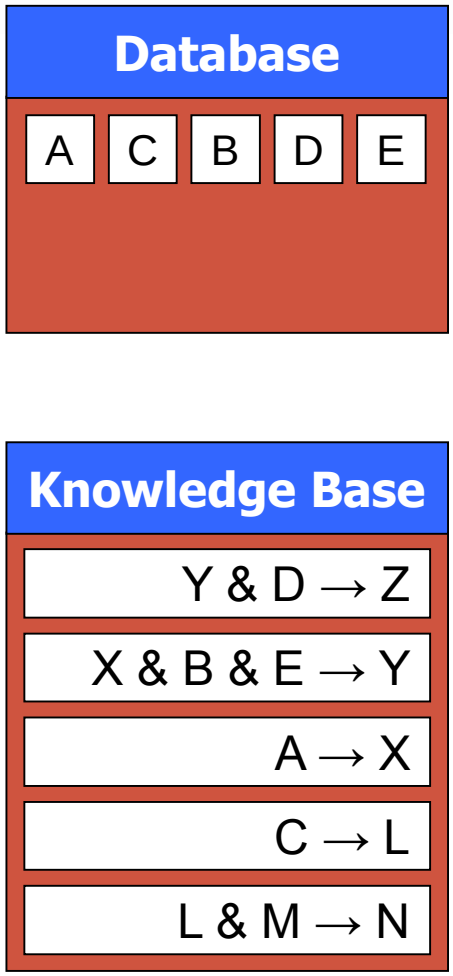


Suppose that we know the facts A, B, C, D, E and the rules shown in the knowledge base to the left

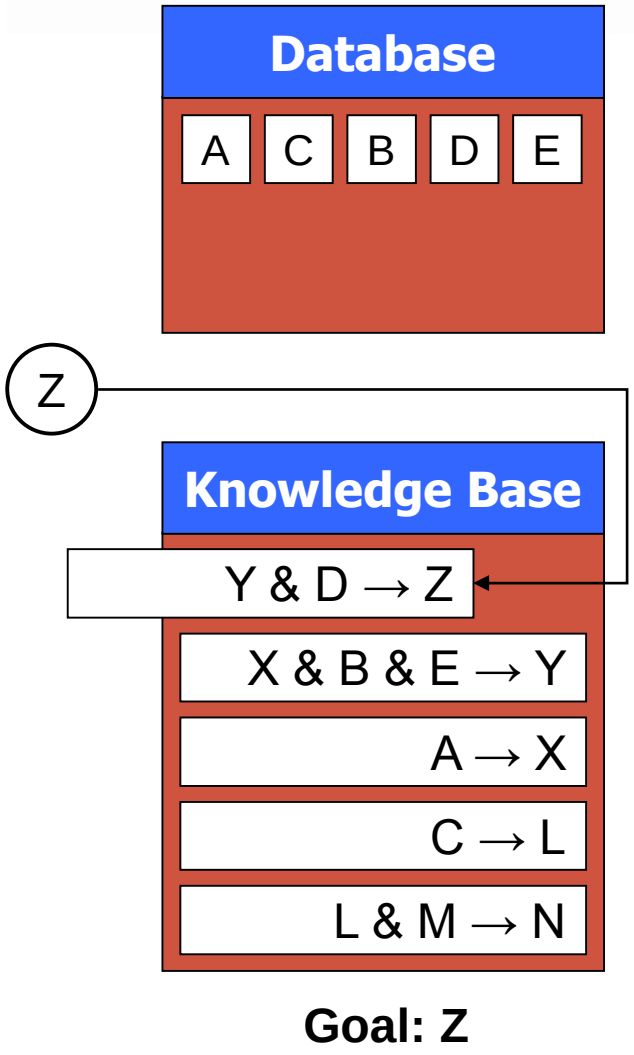
Can we infer the fact Z?



Backward Chaining Example

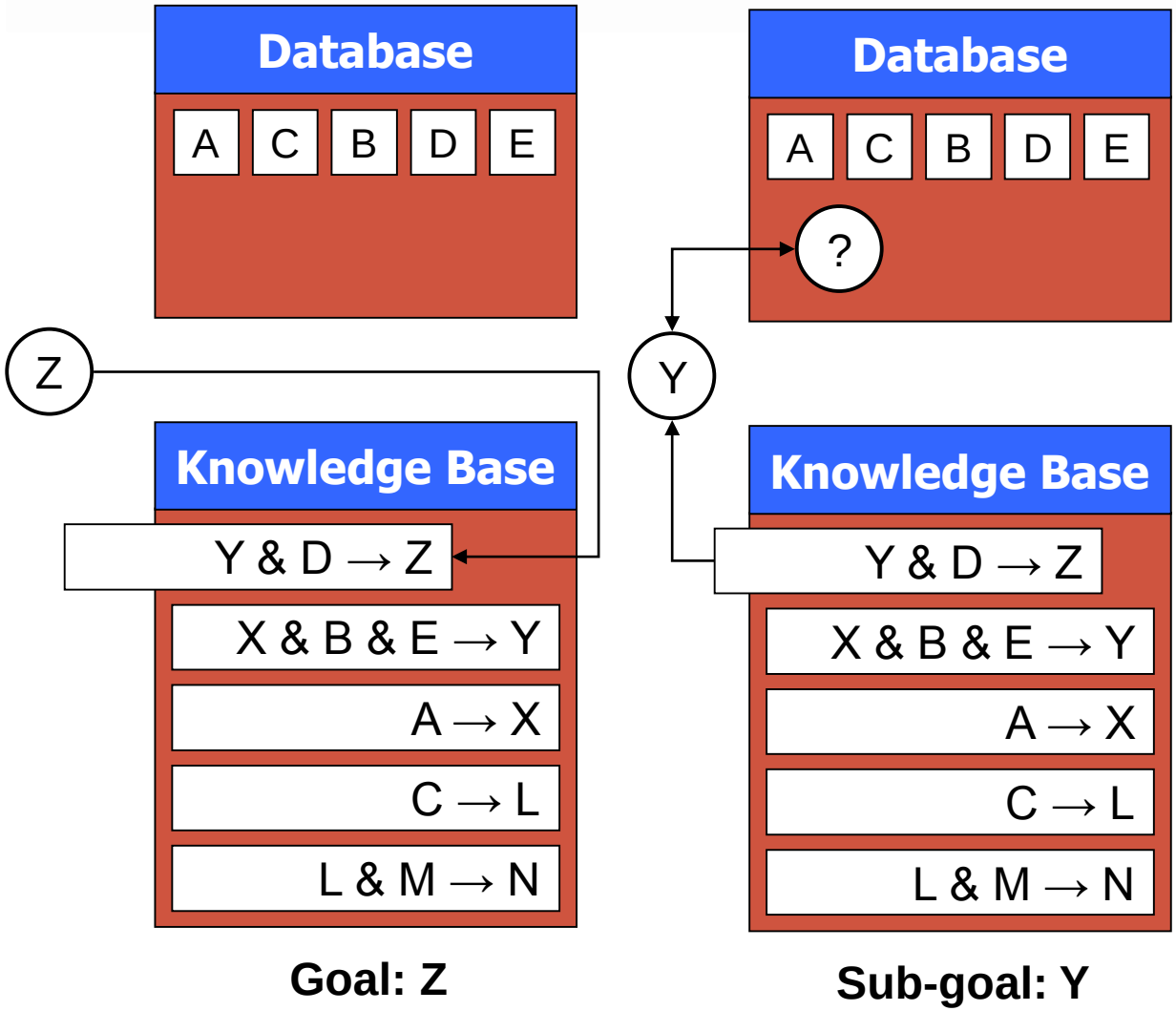


Backward Chaining Example

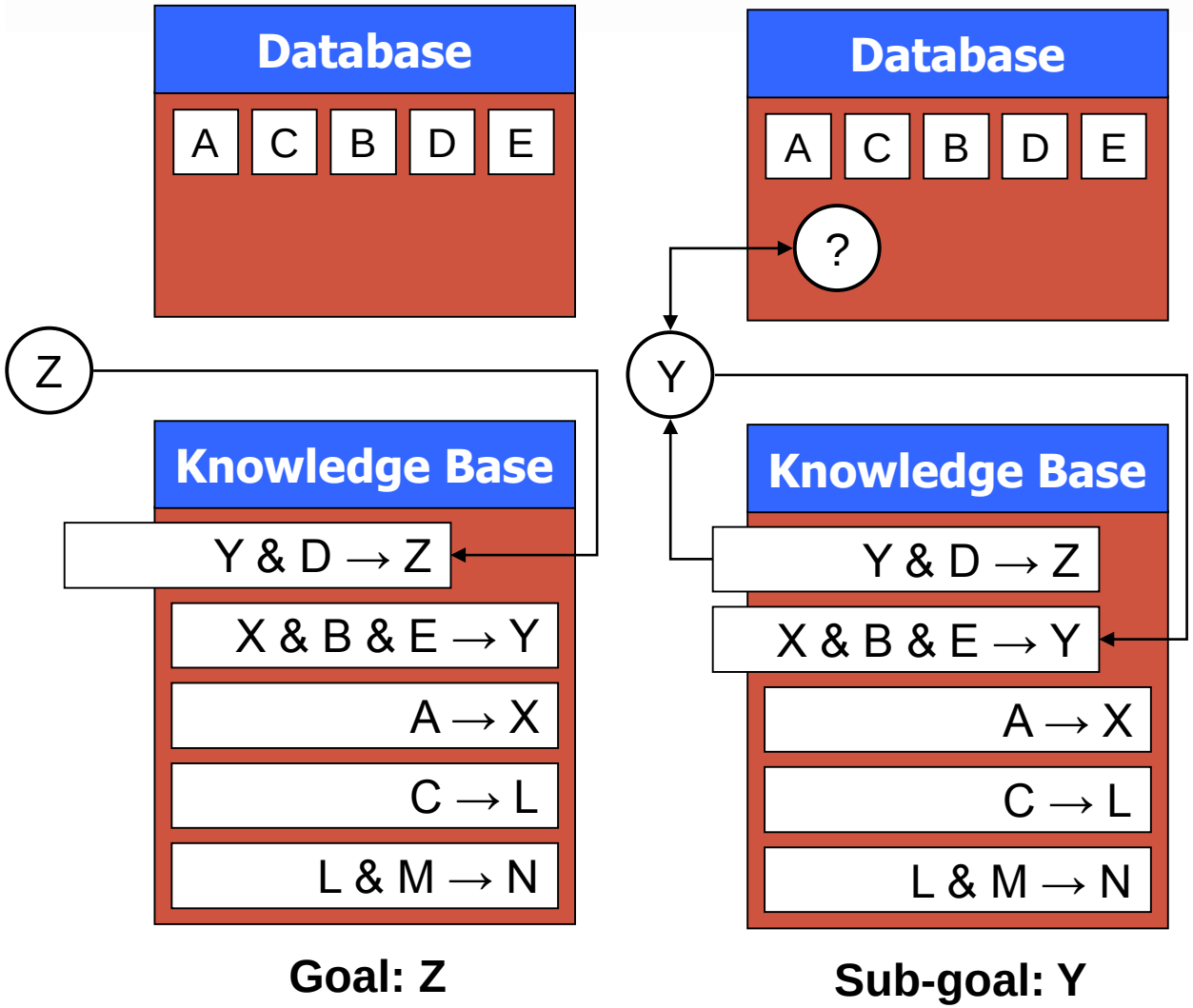


Goal: Z

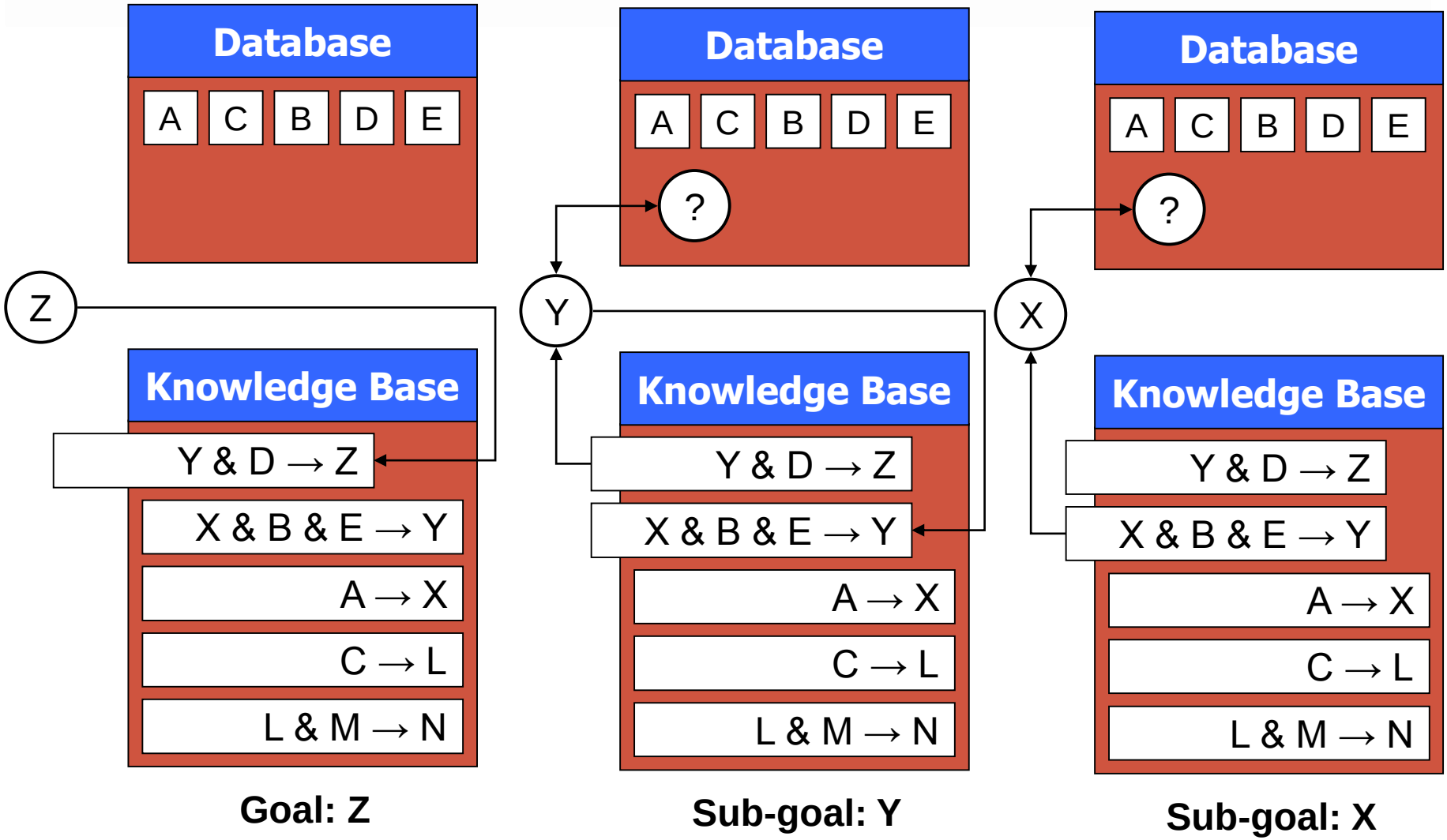
Backward Chaining Example



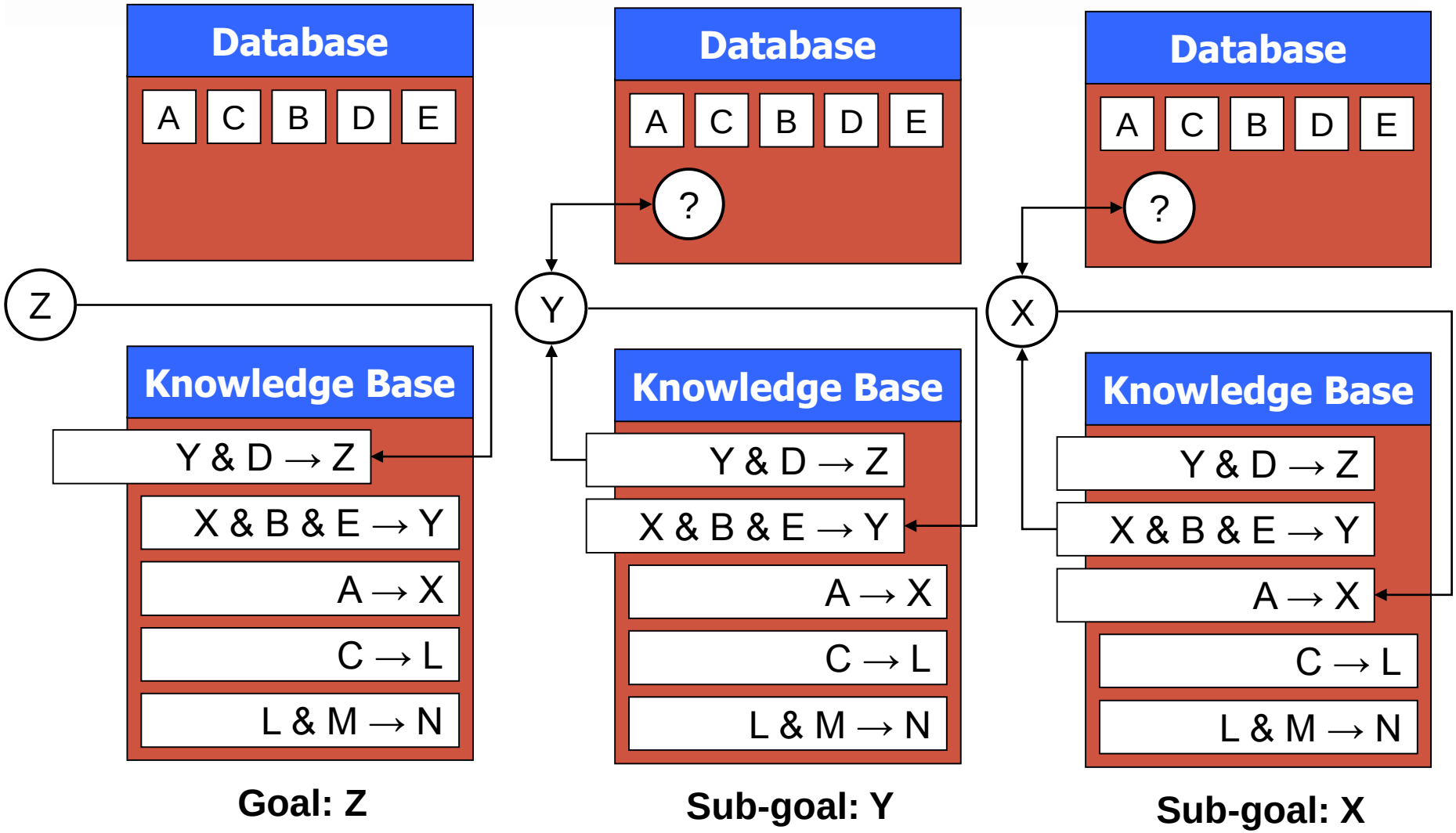
Backward Chaining Example



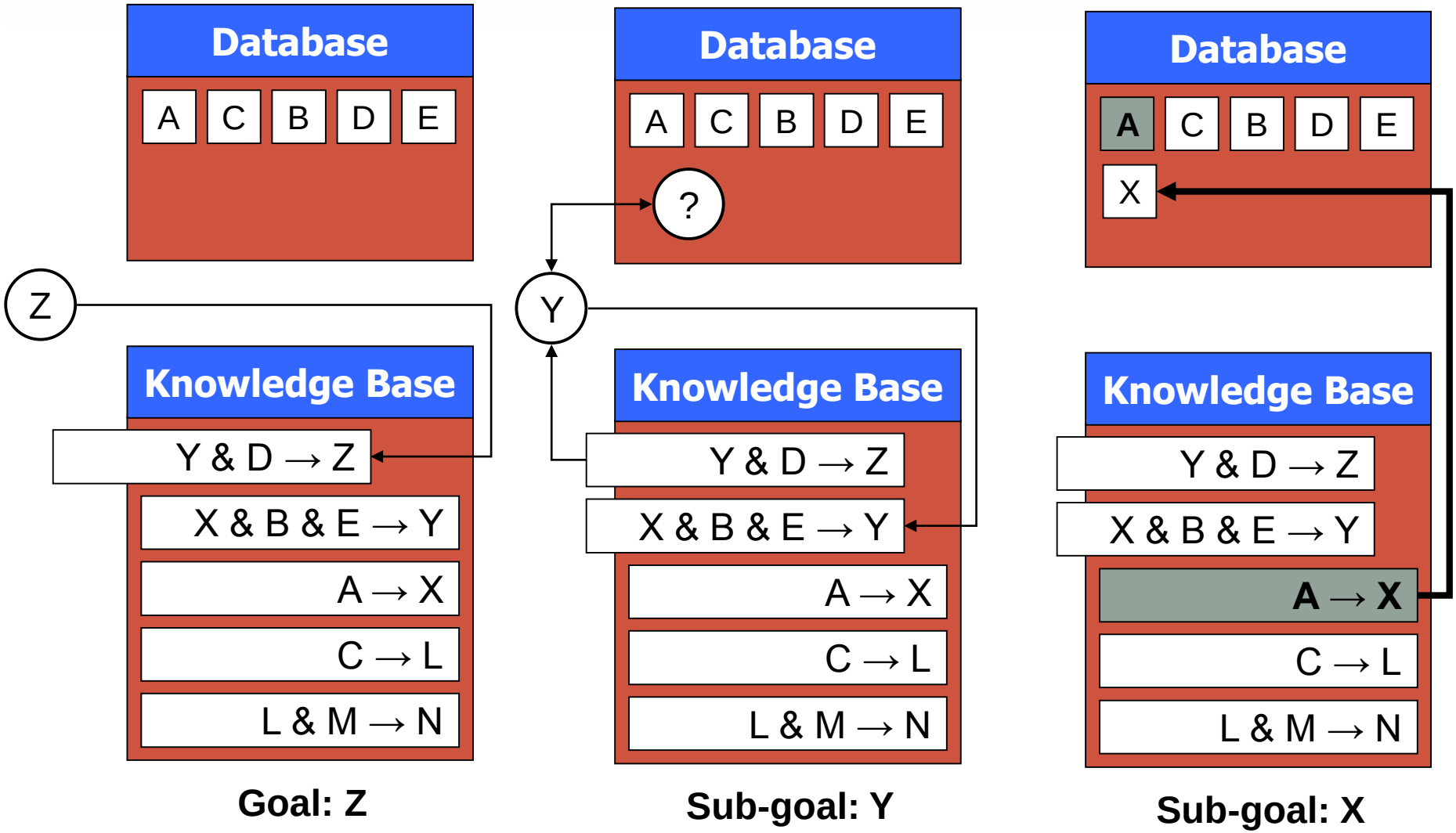
Backward Chaining Example



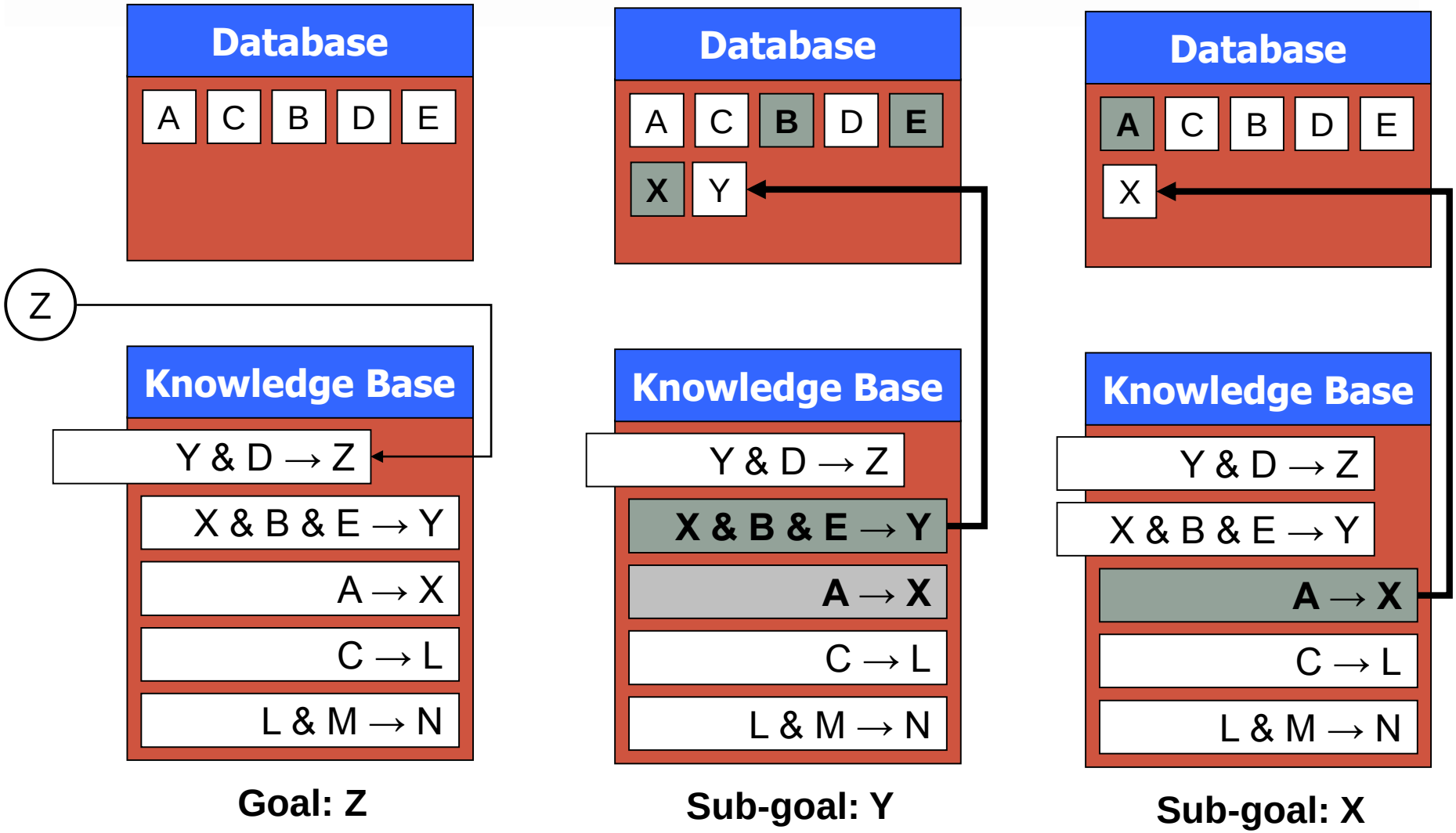
Backward Chaining Example



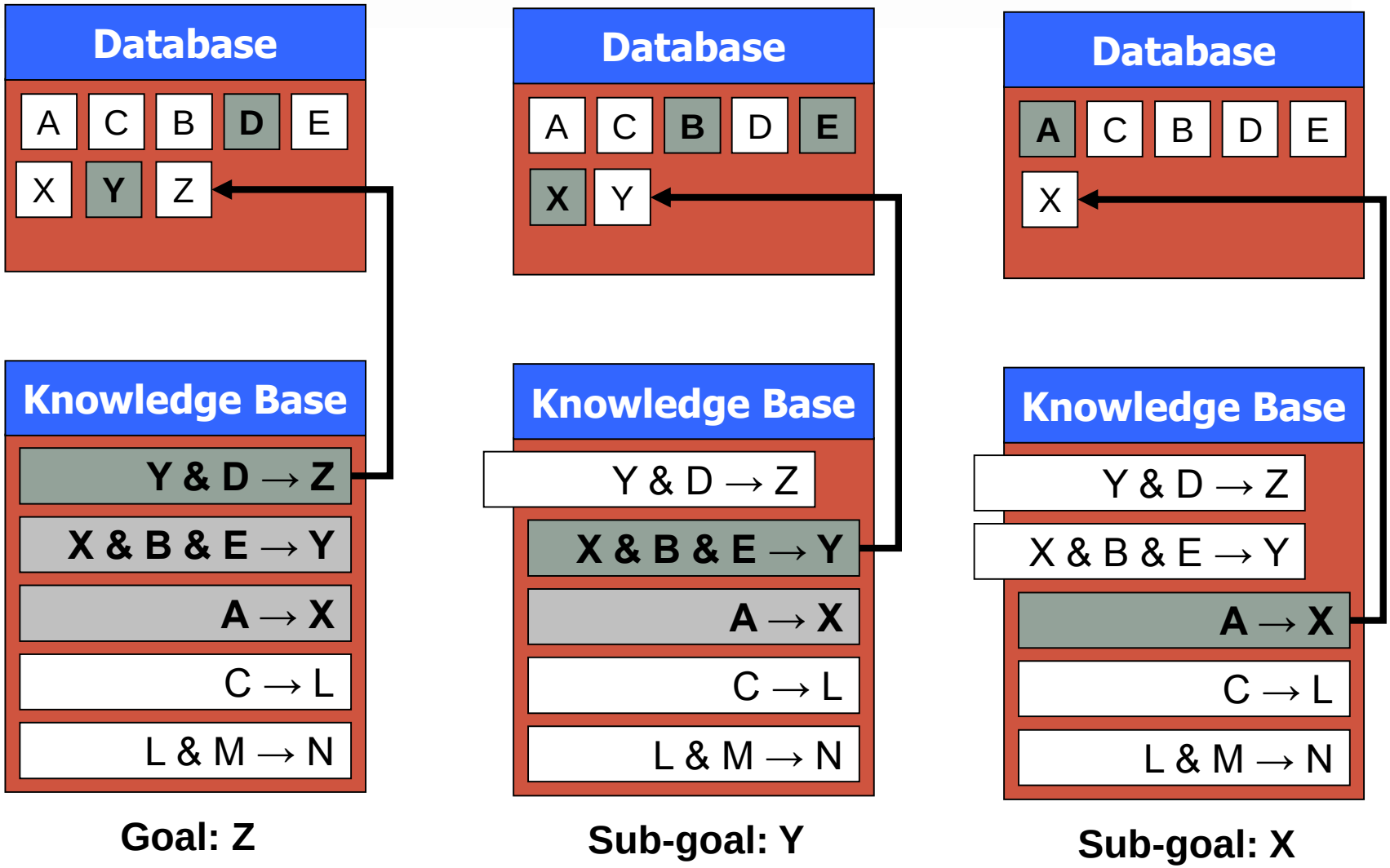
Backward Chaining Example



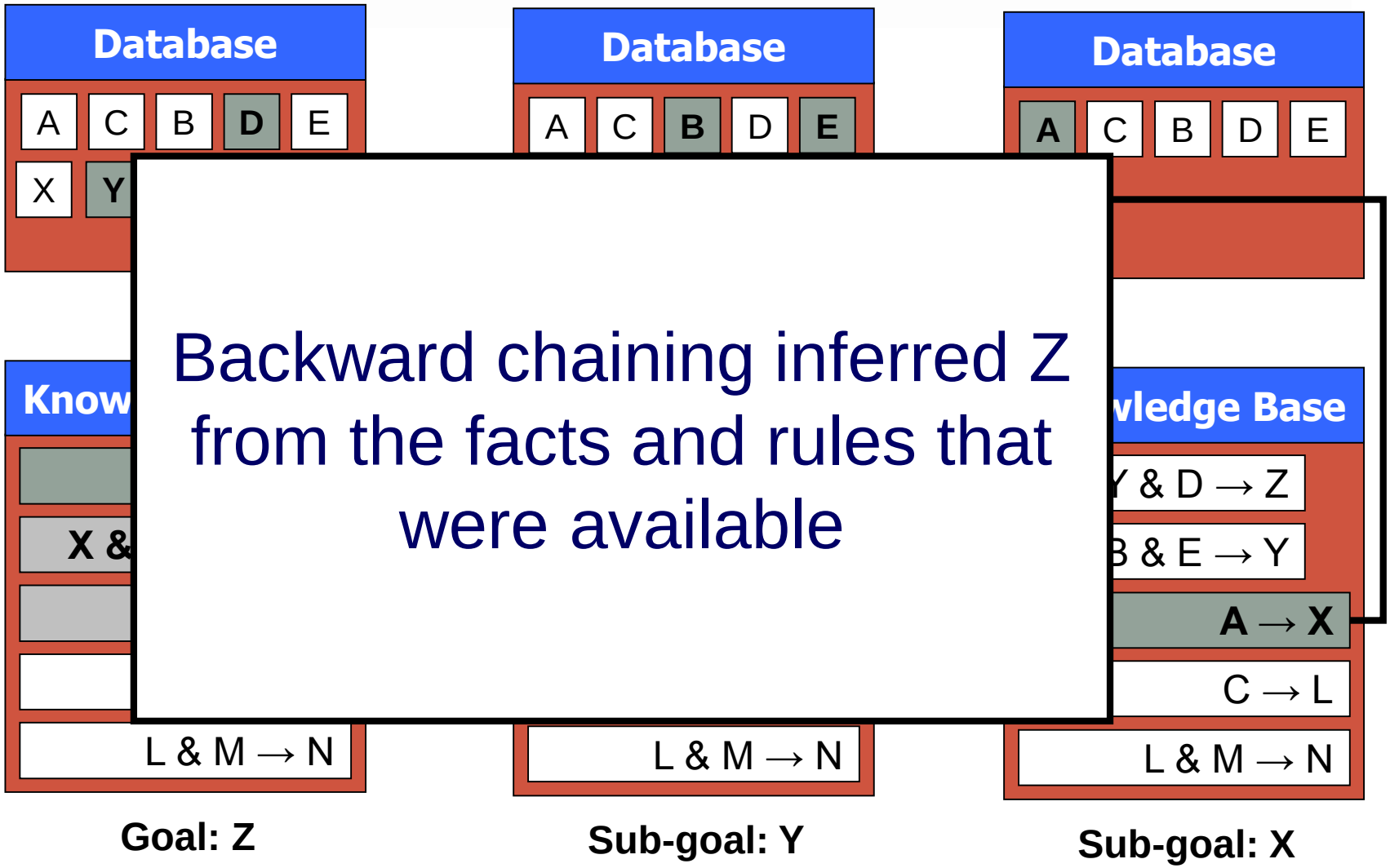
Backward Chaining Example



Backward Chaining Example



Backward Chaining Example



Data Driven Vs. Goal Driven

Data-driven reasoning is appropriate when there exist many equally acceptable goal states, a narrow body of facts and rules and a single initial state

- Required facts are available
- It is difficult to form a goal to verify

Goal directed inference is relevant when

- Relevant data must be acquired during the inference process
- Large number of applicable rules exist
- An obvious goal to verify is available

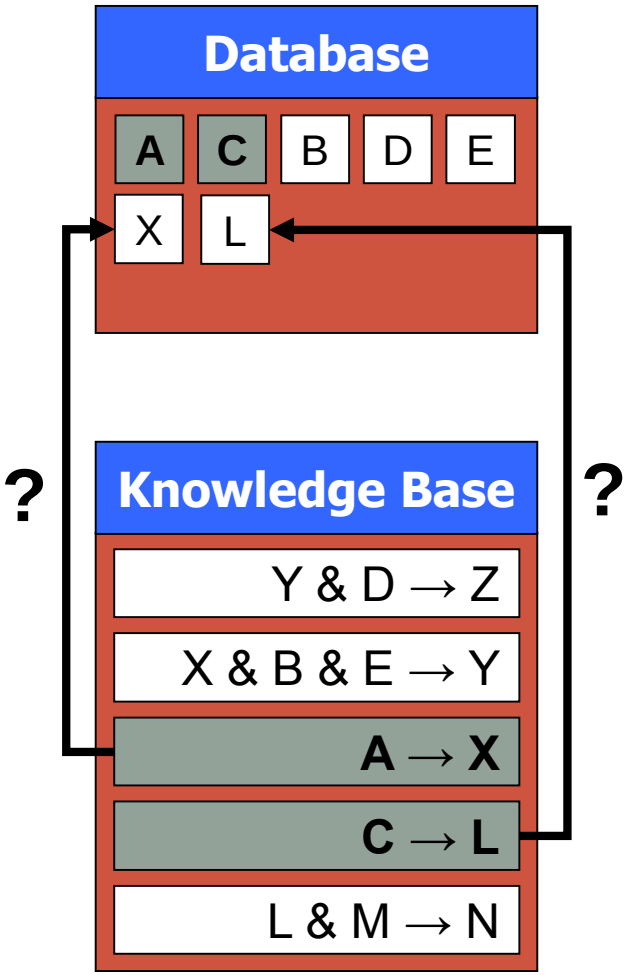
Conflict Resolution

What if more than one rule matches a in a particular situation?

We have actually already seen this in one of our examples

What should we do?

- The answer is referred to as ***conflict resolution***



Conflict resolution strategies

Where more than one rule is applicable, they are said to be in **conflict**. The rules the are in conflict are said to form a **conflict set**.

In response to detecting a conflict set, the inference engine must execute some **conflict resolution strategy** that resolves the conflict and fires (as opposed to triggers) the appropriate rule.

Approaches to conflict resolution

How do you chose which rule from the conflict set to fire?

Rule Ordering

- Choose the first rule in the text, ordered top-to-bottom

Most recent

- Prefers rules that use facts most recently added
- Focuses on single line of reasoning

Specificity

- More specific rules are preferable to more general one
- A rule is more specific if it has more conditions satisfied

Approaches to conflict resolution (cont.)

Refraction

- Once a rule has fired it may not fire again until working memory elements that match its conditions have been modified
- Discourages looping

User-defined

- allow user to prioritize rules

The structure of rules and the conflict resolution scheme used controls the fashion in which the space is searched

Which conflict-resolution strategy to choose also depends on whether you use data-driven or goal-driven inference

Expert systems

An expert system is a computer system that facilitates solving problems in a given field or application by drawing inference from a knowledge base developed from human expertise.

Expert systems aim to capture specialist human expertise which is in short supply.

A rule-based expert system captures the knowledge of an expert in a set of production rules and combines these with observed data to generate inferences

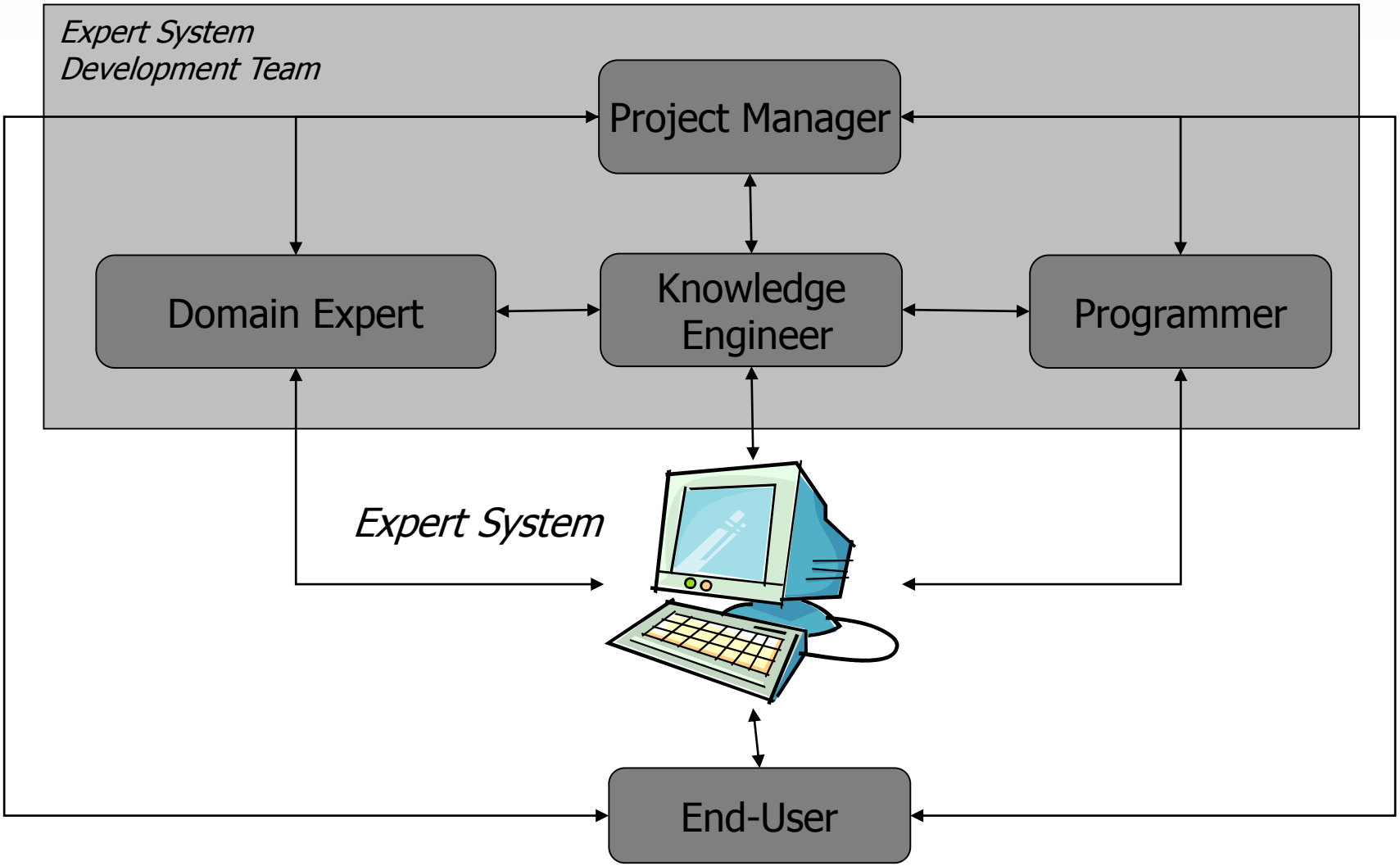
Expert systems

The term **expert system** sometimes is used synonymously with **knowledge-based system**, although it usually emphasizes expert knowledge.

Expert system shell is an expert system with the knowledge (rules and facts) removed

- Example: Jess

Expert Systems Development Team



Development Team

Domain Expert

- Knowledgeable and skilled person capable of solving problems in specific domain

Knowledge engineer

- Capable of designing, building and testing expert system

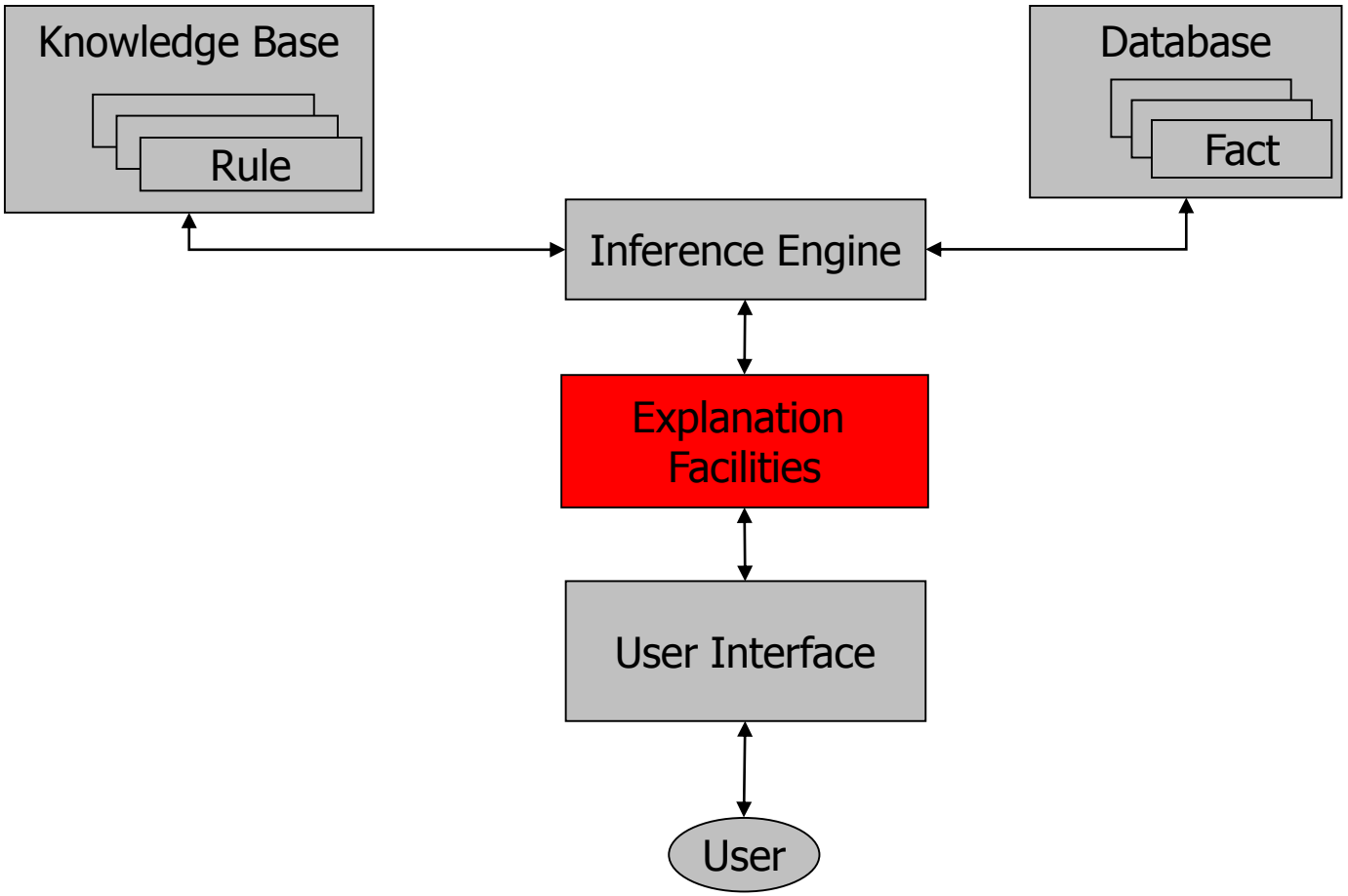
Programmer

- Develop knowledge & data representation structures
- Control structure
- Dialog structure

Project manager

End-User

Architecture of rule-based ES



Characteristics of an ES

Built to perform at human expert level in narrow, specialised domain

- High-quality performance
- Timely solutions
- Use heuristics to guide reasoning

Explanation capability – unique to ES

- Enables system to review reasoning and explain decisions
- Traces rules that have been fired

Can work with incomplete data

Can make mistakes (as can humans)

Knowledge (KB) separated from processing (inference engine)

Comparison

Human Expert	Expert Systems	Conventional Program
Use knowledge in form of heuristics to solve problems in narrow domain	Process knowledge expressed as rules, use symbolic reasoning to solve problems in narrow domain	Process data and use algorithms to solve general numerical problems
Knowledge exists as compiled form in brain	Knowledge & Processing clearly separated	No separation of knowledge & control structure
Capable of explaining reasoning and providing details	Trace rules fired & explain how conclusion is reached and why specific data is needed	No explanation
Use inexact reasoning, can deal with incomplete, uncertain, fuzzy information	Permit inexact reasoning, can deal with incomplete, uncertain, fuzzy information	Only work with complete exact data
Can make mistakes when information is incomplete or fuzzy	Can make mistakes when information is incomplete or fuzzy	Provide no solution when information is incomplete or fuzzy
Quality of problem solving improves with practice & training. Process is slow, inefficient & expensive	Quality of problem solving improves by adding new rules or adjusting old ones. Changes are easy.	Quality of problem solving improves by changing program code – affects knowledge & processing. Changes are difficult.

Expert systems

Sample domains for expert systems

- Medical expertise
- Computer configuration expertise
- Expertise for oil exploration

Aim was to develop systems capturing this expertise, so the knowledge could be deployed where experts are unavailable.

Common approach is to have knowledge represented as if-then rules.

Reasoning strategy could be forward or backward chaining.

MYCIN

MYCIN (Feigenbaum, Shortliffe, 1972-80) at Stanford University, infectious blood diseases.

MYCIN's knowledge base consists of set (~450) of IF-THEN rules, e.g.

IF

the infection is primary-bacteremia

AND the site of the culture is one of the sterile sites

AND the suspected portal of entry is the gastro-intestinal tract

THEN

there is suggestive evidence (0.7) that infection is bacteroid.

MYCIN: Problem solving strategy

MYCIN could use backward chaining to find out whether a possible bacteria was to blame.

This was augmented with certainty factors that allowed an assessment of the likelihood, if no one bacteria was certain.

MYCIN's problem solving strategy is simple:

- For each possible bacteria:
  - Using backward chaining, try to prove that it is the case with some certainty.
- Find a treatment which "covers" all the bacteria above some level of certainty.

MYCIN: Problem solving

The system interacts with the user

- Certain facts are marked as "askable", so if they couldn't be proved, ask the user.

This results in following style of dialogue:

- MYCIN: Has the patient had neurosurgery?
  USER: No.
  MYCIN: Is the patient a burn patient?
  USER: No.
  …
  MYCIN: It could be Diplococcus.

When to consider expert systems

- The need for a solution must justify the costs involved in development.
- Human expertise is not available in all situations where it is needed.
- The problem may be solved using symbolic reasoning techniques (e.g. no requirements for physical skill).
- The problem is well structured and does not require (much) common sense knowledge.
- The problem cannot be easily solved using more traditional computing methods.
- Cooperative and articulate experts exist.
- The problem is of proper size and scope (highly specialized expertise, but would only take a human expert a short time to solve).

Advantages of rule-based systems

- Natural (intuitive) knowledge representation
- Uniform structure
- Separation of knowledge from processing
- Dealing with incomplete or uncertain knowledge

```
IF
  season is autumn AND cloudy AND wind is low
THEN
  forecast is clear     {cf 0.1}
  forecast is drizzle   {cf 0.3}
```

Disadvantages of rule-based systems

- Non-transparent relations between rules
- Ineffective search strategy
- Inability to learn

Advantages of Expert systems

Increased availability

Most expert systems can be used on standard computers. This results in an expert system being a mass production of expertise.

Reduced cost

Expert systems may be used in hazardous environments.

Permanence

Expert systems do not quit, retire, or die – the only drawback is that the knowledge base must be maintained.

Multiple expertise

Expert systems are often developed around the knowledge of many Experts, not just one. In consequence, the level of expertise of the system may exceed the expertise of an individual expert

Increased reliability

Expert systems increase confidence in human decision making by providing a second impartial opinion.

Advantages of Expert systems (cont.)

Explanation.

The expert system can provide explicit detail of the reasoning it used to come to its conclusion – again increases confidence.

Fast response.

Some applications require (near) real-time responses. Computers work consistently quickly, and are always available for problems. Computers don't need to take lunch breaks

Steady, unemotional, and gives a complete response at all times.

Very important in real-time and emergency situations. Human experts can be tired or under stress

Intelligent tutor.

Expert systems can act as intelligent tutors for students by letting the student run sample programs and explaining the systems reasoning for what happens.

Summary

Effective systems have been developed that capture expert knowledge.

Typically combine rule-based approaches, with additional certainty/probabilistic reasoning.

Questions?